THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 53, No. 8, October 1974
Printed in U.S.A.

LAMP:

Controllability, Observability, and Maintenance Engineering Technique (COMET)

By H. Y. CHANG and G. W. HEIMBIGNER

(Manuscript received February 28, 1974)

A new technique has been developed for organizing (or reorganizing) system design to enhance fault diagnosability. This technique is called the controllability, observability, and maintenance engineering technique, or COMET. Using graph-theoretical analysis, one can systematically apply COMET to a proposed or an existing digital system to determine the placement of control, access, and monitor points for diagnostic testing. In addition, it provides a means of studying the trade-offs between fault resolvability and the cost of maintenance hardware and/or packaging.

COMET offers an orderly approach to implementing an overall diagnostic design by providing guidelines in early design stages. A design developed using COMET has the following advantages: trouble location manual data can be generated without the use of fault simulation, multiple faults and/or nonclassical faults are locatable if they are detectable, and diagnostic or trouble-location information can be easily updated in accordance with hardware changes. Studies indicate that applying COMET to an existing processor design would require a modest increase in hardware of less than 10 percent.

I. INTRODUCTION

Recent advances in integrated-circuit technology offer the circuit and system designers many opportunities to explore new, low-cost, high-performance design techniques. The increased operational speed and the logic complexity of many medium-scale-integration (MSI) and large-scale-integration (LSI) designs, however, also present acute problems in factory testing and field maintenance. For

factory testing, it becomes increasingly difficult to diagnose faults in an LSI package, partly owing to equipment packaging constraints and partly to inadequate fault isolation tecbnique(s) for nonclassical and/or multiplc faults. In field maintenance, while the isolation of faults to a component or chip level is unimportant, the problem of quickly, and automatically, detecting and recovering from faults is further compounded by increases in circuit size and complexity. In addition, tbe cost of using fault-simulation techniques to generate data for the trouble-location manual[1] (TLM) may become economically prohibitive, especially for large systems. Finally, tbe problem of accurately updating trouble-location data whenever circuit or design changes occur remains important but unresolved.

Many designers of fault-tolerant computer systems bave studied tbese problems.[2] Some have proposed design approaches with built-in automatic-fault-detection bardware.[2,3] Others bave explored diagnosable design concepts purely from a structural standpoint based on graph-theoretical techniques.[4,5] Unfortunately, the search for practical methods of generating (and updating) TLM data for large systems bas been largely unsuccessful.

This paper describes a technique called COMET (controllability, observability, and maintenance engineering technique) for organized system design and system reorganization to enhance diagnosability. COMET enables a designer to systematically establish the diagnosability of a system by combining circuit design, physical arrangement, and maintainability considerations. It also offers an efficient and practical way to generate TLM data.

In Section II, tbe concept of COMET is described, followed by a detailed discussion of the tecbnique and its relation to fault location. Possible methods of implementation are tben discussed, along witb the results of applying tbis tcchnique to a small self-cbecking processor.[3] Lastly, tbe long-term impact of COMET on system design is pointed out.

II. DESCRIPTION OF CONCEPT AND TECHNIQUES

2.1 *Philosophy and characteristics*

Tbe design of a fault-location procedure involves several steps. For a given processor or circuit, a set of tests capable of detecting all tbe assumed faults is first derived. Tbe usual assumptions are tbat faults are solid and are of the "stuck-at" type. This step is called tbe test-derivation phase. These tests are tben verified eitber by sample fault simulation or by a complcte fault simulation to determine if tbey are

indeed a good set of tests. The next step is to derive for each fault in the fault set the corresponding test results. This is done by simulating each fault with respect to the tests that are designed to detect this fault. The test results ohtained hy this technique are then processed to form a TLM. These two steps are called TLM data generation and data processing.

Suppose the processor has only one circuit pack. Every time a fault occurs and is detected, this circuit pack is replaced. It will no longer be necessary to distinguish faults in this processor; the fault-location problem is eliminated and the TLM data-generation and processing steps disappear. The only step required is to derive the test capable of detecting all faults and to record the test results of the "good machine."

Now if the processor is composed of more than one circuit pack, the following conceptual approach may be used. At the heginning of diagnosis, half of the processor is disahled. This can he done, for example, hy physically removing half of the circuit packs in a processor. Diagnostic tests are run only on the enahled portion and only the pass-or-fail data of the tests are recorded. Thus, if a fault exists in the enahled portion of the processor, the test result will give a failure indication, meaning that the fault is not in the disahled portion. However, if the test result gives a pass indication, this means that the fault is in the portion that has heen disahled or removed.

Based on the pass-or-fail indication, one can further partition the enahled portion (in the case where the fault is in the portion that remained enabled during testing), or the disahled portion (in the case where the fault is in the disahled portion) to allow further testing. A general flow diagram of this procedure is shown in Fig. 1. Disahling means that the circuit packs associated with the disahled portion are

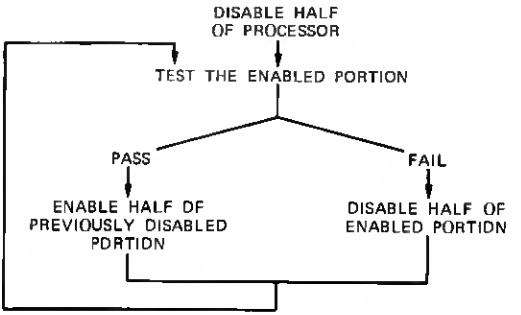


Fig. 1—Flow diagram of disabling process.

either physically disconnected from the processor or are logically in a passive state.

Figure 2 shows an example of how faults are located using this technique in a processor composed of four circuit packs (A, B, C, and D). Assume that the fault is in circuit pack B. Diagnostic tests are first performed with circuit packs C and D disabled or removed. The tests, which are designed to detect all the faults in packs A and B, are run and a failure is indicated. It can then be concluded that some failure exists in either pack A or B. Next, the diagnostic tests are run on circuit pack A with circuit pack B disabled. This time, however, because the fault (which is in circuit pack B) has been masked, the test result will show a pass indication. It can then be concluded that a fault is in circuit pack B because it gives a fail and then a pass signature. In other words, by successively reducing the circuitry under test and hy only recording the fail/pass results in each step, the locations of all faulty circuit packs are uniquely identified.

It is quite apparent that the basic difference between this technique and the conventional approach is that in the latter one must, for isolation purposes, distinguish the faults not only from the good machine but also from every faulty machine. In other words, the important consideration is where the fault is. In the proposed approach, however, we are not required to distinguish the various faulty machines; we are only interested in whether the test result from the good machine differs from that of the faulty machine. Resolution is obtained by successively reducing the circuitry under test.

For each element of the partition (e.g., packs A and B of the first partition in Fig. 2, or circuit pack A of the second partition in Fig. 2),

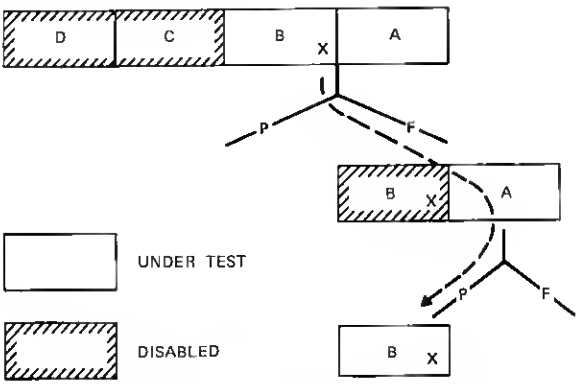


Fig. 2—Proposed method of fault location.

only a pass or fail indication is required. This means that the test result of the good machine for each partition is all that is required; there is no need to simulate faults to generate sufficient information for the distinguishability of the faults within the element of partition. It can be seen that each faulty circuit pack is identifiable by a unique pattern of pass-and-fail indications; the pass-and-fail numbers are in a one-to-one correspondence to the circuit packs in the processor. This means that the number of trouble numbers in the TLM is drastically reduced.

Another characteristic of the proposed technique is that multiple faults on a circuit pack are locatable as long as they are detectable by the applied tests. This should improve accuracy since the requirement for consistency of test signatures for TLM lookup no longer exists.

2.2 Description of techniques

2.2.1 Controllability and observability

There are some problems that must be solved. First, one must design a disabling process that allows circuit packs to be selectively disabled or removed from the processor. Second, *controllability* of the various circuit packs or functional blocks must be established. For example, as shown in Fig. 3a, gate G' must be operational to test gate G. That is, G' must not be disabled when G is being tested. If, however, G' is in the portion that has been disabled, the testing of G and therefore the test results become meaningless because G is not controllable from G'.

Similarly, a proper ordering of the various circuit packs or functional blocks in relation to *observability* must be derived. As shown in Fig. 3b, to observe the test results of a fault (marked ×) associated with gate G, the output of G must not be in the logic block that has been disabled. Otherwise, the test results will show an all-tests-pass

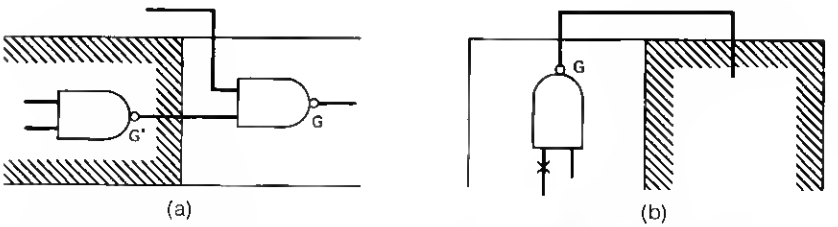


Fig. 3—(a) Controllability-ordering relations. (b) Observability-ordering relations.

outcome even though tbe fault is present on gate G. Thus, the necessary condition for this tecbnique is the establishment of an ordering relation of controllability and observability such tbat a partitioning procedure can be used for fault isolation. If the elcments of partition i do not depend on elements of partition j (wbere $j > i$) for controllability and observability, one can successively apply thc partitioning and testing process to locate the faulty circuit pack, providing that the disabling technique of circuit packs is available.

2.2.2 Logic-disabling technique

Tbere are a number of disabling techniques possible. Tbe first and tbe most obvious one is to pbysically remove tbe circuit pack(s) from tbe processor. This is only feasible if a reliable connector is available and tbe number of circuit packs involved is small. An alternative is to pbysically disconnect the input and output leads of a circuit pack by some mecbanical device. A tbird alternative in some cases is simply to remove tbe power and ground leads of a circuit pack, thus putting the circuit pack in tbe passive state. A fourtb alternative is the logic-disabling technique illustrated in Fig. 4.

If control lead C_i (Fig. 4) goes to 0, it forces output of the output gates to logical value 1, whicb for NAND logic is the passive state. After tbe circuit pack bas been disabled, error symptoms caused by any fault or faults in tbe circuit pack (marked ✕ in the illustration) cannot propagate beyond the circuit-pack outputs.* Tbus, if eacb circuit pack i is modified by adding disable control lead C_i, eacb circuit pack in tbe processor can be enabled or disabled selectively.

2.2.3 Partitioning techniques

Once a practical way of disabling circuit packs is obtained, tbe next step is to devise a tecbnique of ordering the circuit packs or tbe functional blocks based on tbe observability and controllability relations. The controllability and observability relations can best be understood by the example sbown in Fig. 5.

We define a functional node as a functionally well-defined logic circuit, such as a rotate circuit, an adder, etc. In some instances, a functional node is also defined as a logically or physically related block of circuitry, sucb as bits 0 through 7 of tbe X, Y, Z registers. To test a functional node, one must apply signals via control inputs and observe

* Note that the stuck-at-0 faults on the disabled gates can still propagate. Treatment of these faults is discussed in Section 2.2.4.

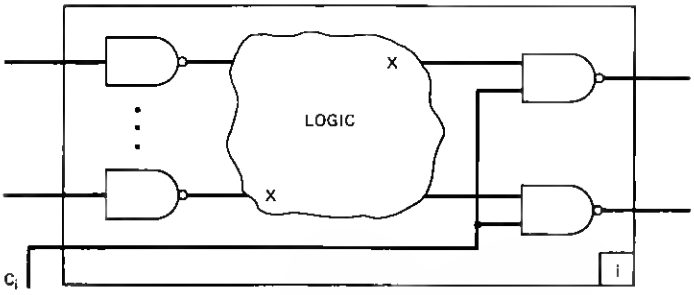


Fig. 4—Circuit-pack logic-disabling process.

test results via some observable outputs. Thus, in Fig. 5, it can be concluded that to test functional node R, the inputs are obtained from functional node A, and control is obtained from functional node C. Functional nodes A and C, therefore, must be operational when functional node R is being tested. Similarly, to observe the test results of R, functional nodes O_1 and O_2 are used. In other words, functional nodes A and C *control* R, and functional nodes O_1 and O_2 *observe* R; they must not be in the portion that is disabled when node R is being tested.

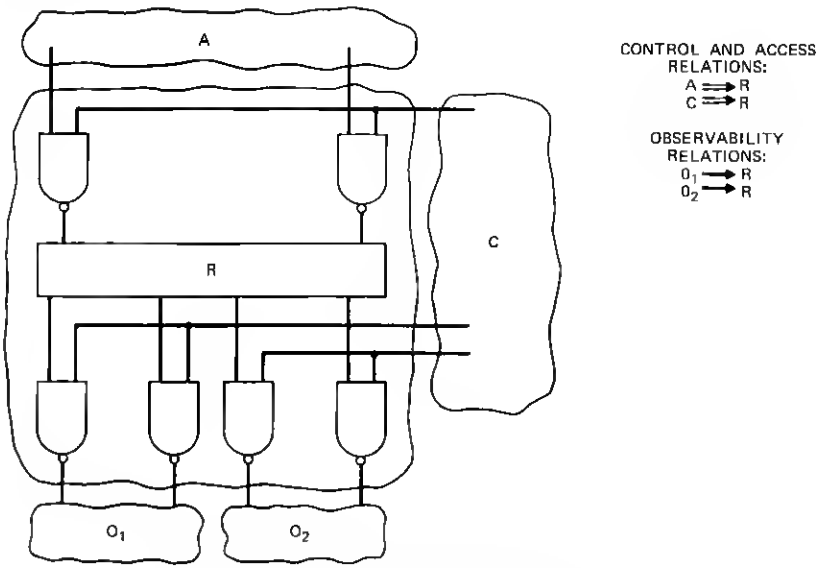


Fig. 5—Controllability and observability relations.

A set of relations can now he defined as follows: node A *controls* node B, if node B requires control from A to he fully testahle. This relation is expressed hy $A \Rightarrow B$. Similarly, node A *observes* node B, if node A is required to ohserve the test results of node B. This is represented hy $A \rightarrow B$.

From a set of properly defined functional nodes, the controllahility and the ohservahility relations among neighhoring nodes can he ohtained. These relations are conveniently represented hy a directed graph where the nodes of the graph correspond to thc functional nodes and the edges of the graph correspond to the controllahility and/or the ohservahility relations. If the resultant graph is loop-frec, all the nodes can he arranged in a partially ordered list so that the higher-order* nodes can he fully tested independently of any lower-order node(s). This guarantees a relation of controllability and ohservahility among all the functional nodes such that we are able to partition the system. However, if the resultant graph is not loop-free, a conflict exists from a controllahility and ohservahility viewpoint.

For example, if some functional node F_6 controls functional node F_1, which in turn controls F_5, and if F_5 ohserves F_6, then a loop exists containing F_6, F_1, and F_5. This loop must he "hroken" to test these nodes. In this context, hreaking loops means that additional control or ohservation must he added in appropriate places to obtain a loop-free graph.

The process of systematically ordering thé nodes and identifying conflicts in the directed graph makes use of graph-theory techniques.[4] The directed graph of the controllahility and ohservahility characteristics of functional nodes can he represented hy a connectivity matrix $C = [c_{ij}]$, where $c_{ij} = 1$ if there is a directed edge from i to j. In other words, if node i controls node j, the entry $c_{ij} = 1$ in the connectivity matrix for the controllahility relation. Similarly, if node k observes node p, the entry $c_{kp} = 1$ in the connectivity matrix for ohservahility.

A node j is *reachable* (controllahle or ohservahle) from node i if and only if there is at least one directed path from i to j. A graph is *strongly connected* if and only if every node is reachahle from any other node. This means that in a strongly connected graph, every node is in at least one loop. A *maximal strongly connected* (MSC) suhgraph is one that includes all possihle nodes that are strongly connected with each

* The term *higher* refers to the location in a diagnostic procedure. The higher nodes are verified first.

other. This designates a maximum set of functional nodes that are in conflict from a controllability and observability viewpoint. A *link* subgraph of a graph is one that contains no strongly connected subgraphs or unconnected subgraphs in it. The link subgraph is loop free and therefore all the nodes in it can be arranged in a partially ordered list.

Generally speaking, all functional nodes are either in some MSC subgraph or in some link graph from the observability and controllability viewpoint. The objectives are, therefore, to locate the MSCs (i.e., areas of conflicts) in the directed graph and to add additional controllability or observability points in order to break the MSCs and arrive at a partial ordering of the nodes. The following is an algorithm for performing this function.

(*i*) Construct connectivity matrix of the functional nodes of the processor with respect to the controllability and observability relations.
(*ii*) Locate all MSCs and represent them as pseudonodes.
(*iii*) Establish the order of the nodes in the new directed graph.
(*iv*) Any MSCs left? If yes, go to step (*v*). If no, exit.
(*v*) Pick an MSC of the highest order and apply MSC breaking technique; then return to step (*iii*).

The connectivity matrix is arrived at by merging the connectivity matrices of the controllability relations and observability relations.

All MSCs found in the directed graph are located and temporarily identified as pseudonodes so that the ordering process of step (*iii*) can be carried out. Ordering means that all nodes having only primary inputs are considered to be of the highest order (i.e., first order), and a node is of ith order if all of its inputs are of order $i - 1$ or less and at least one of the inputs is of order $i - 1$. In an ordered list, nodes of the ith order do not depend on those of jth order, for $j \geqq i$ for controllability and observability. The kth order is said to be higher than the ith order if $k < i$. Once the ordering process is performed, we then proceed to break the MSCs, if necessary.* The process of breaking MSCs is similar to the one described by Ramamoorthy.[4] First, the entry nodes of a given MSC are identified. An entry node having the highest ratio of number of incoming edges to number of outgoing edges is selected. All edges entering it are deleted; this means that additional

* This is necessary only if the required resolvability is one faulty circuit pack and there exist some MSCs that cannot be packaged on one circuit pack.

control and/or monitor points are added to those nodes associated with these edges.

The result of this process [steps (*i*) through (*v*)] is a loop-free directed graph or a partially ordered list of nodes. Nodes of the *i*th order are completely testable using only those nodes of orderings higher than *i*. At this point, packaging considerations can be incorporated in order to arrive at a reasonable set of partially ordered lists of circuit packs. The general guide-lines for packaging are:

(*i*) Group only nodes of the same ordering on one circuit pack or set of circuit packs.
(*ii*) If this is not possible, then group a node (or a set of nodes) of the *i*th order with those of order $i - 1$ or $i + 1$, but not both.

These guidelines assume that the resolution is to be to one circuit pack. For example, the groupings of functional nodes shown in Figs. 6a and 6b are acceptable. In Fig. 6b, the ordering of circuit packs P_j and P_{j+1} are irrelevant because the two circuit packs are equivalent. But the ordering of circuit packs P_{j-1} and P_j or P_{j-1} and P_{j+1} is im-

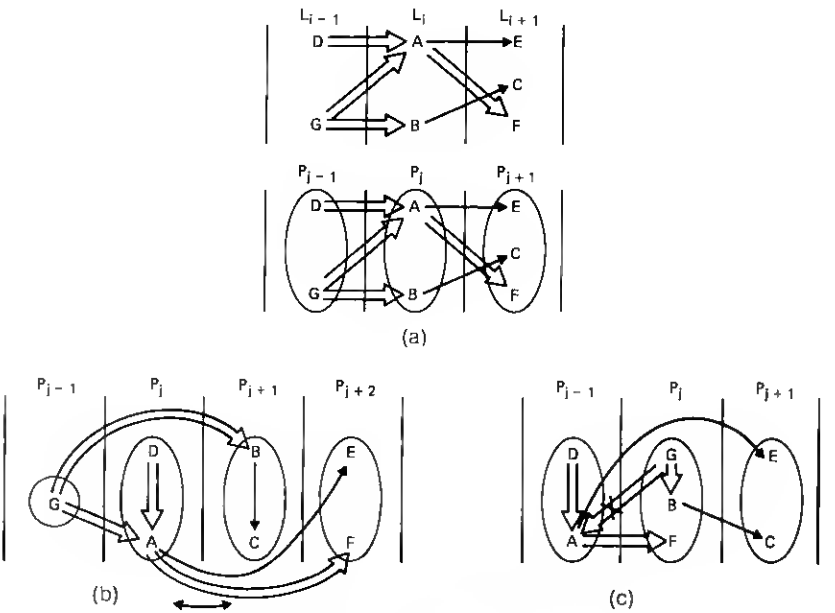


Fig. 6—Packaging considerations.

portant. This is because functional node G controls both nodes A and B, and therefore G must be of a higher ordering than nodes A or B. Figure 6c illustrates a conflict because G controls A but A controls F. Suppose functional nodes G, B, and F are packaged on one circuit pack. A conflict then exists because packs P_{j-1} and P_j cannot he separated for testing purposes. In other words, the resolution has been degraded from one pack to two packs in this case.

Once tbe nodes have been packaged and all the packaged circuits have been ordered, a partitioning procedure can be applied to the packaged circuit packs. The following example shows the process of ordering and partitioning of nodes.

Example: Suppose the directed graph shown in Fig. 7a represents thc controllahility and the observability relations of a circuit. Each node in thc graph represents a functional entity; each edge represents either a controllability relation (denoted by $\Rightarrow$) between the two nodes or an observahility relation (denoted by $\rightarrow$) between the two nodes. For example, node D observes node E; node E controls node H. The information represented by this graph is equivalent to a connectivity matrix which can be constructed by examining each functional node in the circuit and its observability and controllability relations with its neighboring functional nodes.

To arrive at a partially ordered list of nodes, the first step is to locate all MSCs and represent them as pseudonodes. In this case, there is one MSC (as indicated by the dotted line in Fig. 7a) denoted by υ. The reduced graph is then ordered by applying the ordering process. For example, to completely test and observe node C which is of order L_2, it is only necessary that nodes of higher order, i.e., nodes A and B of order L_1, be availahle for control and observation.

If all the nodes can be packaged at this point according to the previous guidelines, a partial ordering of nodes has been obtained. However, if, for example, the pseudonode υ contains too many functions to be packaged, the pseudonode υ must be further decomposed by breaking the MSC it represents. The entry nodes to this MSC are nodes D and G. Node G is chosen and the edge D–G is broken by adding control to node G. At this point there is still another MSC in the graph so the process is repeated (see Fig. 7b).

A new MSC denoted by a pseudonode υ_1 is identified, and the graph is ordered once again. The MSC denoted by υ_1 is again "broken" after having decided to add an observable point to the functional node D. The final ordered list of functional nodes is shown in Fig. 7c; these

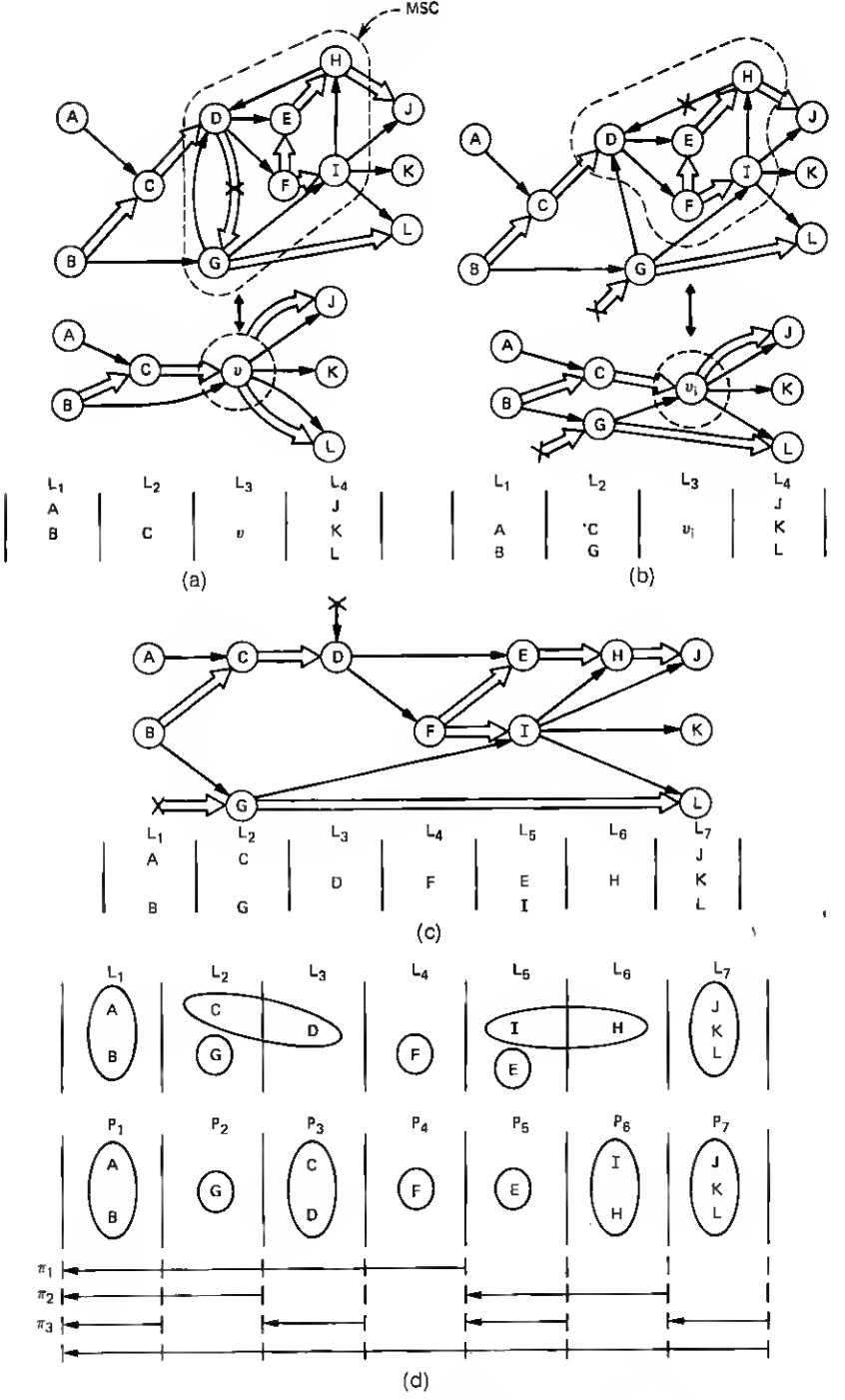


Fig. 7—Partitioning and ordering of functional nodes.

nodes form a partially ordered list in that nodes of order L_i are testahle using only those nodes of an ordering higher than L_i.

To demonstrate how fault(s) can be isolated using this process, the functional nodes are assumed to he packaged as shown in Fig. 7d. Without loss of generality, a hinary partitioning process will he used in the following cases.

Case 1: Single-fault location. Suppose a fault occurs on circuit pack P_5. Diagnosis will he performed first on portions of circuitry consisting of P_1, P_2, P_3, and P_4, while circuit packs P_5, P_6, and P_7 are disabled. The test results are valid hecause the testing of circuit packs P_1 through P_4 does not depend on circuitry associated with circuit packs P_5 through P_7 due to the proper partial ordering requirement. The first test result ($\pi 1$) will yield a pass indication implying that the fault is not on circuit packs P_1, P_2, P_3, or P_4. Following the process discussed in Section 2.1 (see Fig. 1), the diagnosis ($\pi 2$) is then performed on circuit packs P_5 and P_6, using circuitry associated with P_1 through P_4, which has heen previously verified, while disahling circuit pack P_7. This time the tests will show a fail indication; the failure is isolated to circuit packs P_5 and P_6. In the next step ($\pi 3$), circuit packs P_6 and P_7 are disahled while running tests on circuit pack P_5 using circuitry associated with P_1 through P_4. The test result again shows a fail indication and, thus, identifies the faulty pack to he P_5.

Case 2: Multiple-fault location. Now suppose that a fault exists on circuit pack P_5 and another fault exists on circuit pack P_3. The initial diagnosis of partition $\pi 1$ shows that a failure is in circuit packs P_1 through P_4 with circuit packs P_5 through P_7 disahled. The failure symptom caused hy the fault of P_5 will not interact with the testing of P_1 through P_4 hecause it has heen disahled.

Next, circuit packs P_1 and P_2 are tested in partition $\pi 2$; this test indicates a pass indication. The final test ($\pi 3$) is on P_3 using circuits associated with P_1 and P_2, and disabling circuit packs P_4 through P_7. This test identifies the fault on P_3. Once this faulty circuit pack is identified and replaced, a complete check is run. The presence of the fault on P_5 will now indicate a test failure. The diagnostic process described previously is repeated again to isolate and identify the second fault.

It can he seen that this process enahles us to systcmatically isolate faults one at a time until all faults in the circuits are identified and repaired. Any fault that gives a test result that is different (regardless of the nature of the fault) from the true-value signature is detectahle. In other words, the single-fault assumption and the classical stuck-at-0 and stuck-at-1 assumptions on failure modes are no longer necessary with this approach.

2.2.4 Global Feedback

The constraint of heing ahle to fully disahle the outputs of *all* circuit packs is an overly restrictive condition. The only reason for disabling is to prevent propagation of fault symptoms *into* the portion of the machine under test. Thus, at any partition only those leads crossing from the lower-order levels into the higher-order levels are of major importance. In a general case, this should represent only ahout half of the leads crossing the houndary. In a machine organized using COMET, it could he expected that the portion of leads crossing from low levels to high levels would he less than half, reflecting the attempt COMET makes to hreak things into a tree structure.

For the purpose of discussion, "global feedhack" will be defined on the linear ordering of functional nodes. A global feedback is any directed *wire* going from a lower-order node to a higher-order node in the list of partially ordered functional nodes shown in Fig. 8. The distinction hetween a wire and a controllahility/observahility edge is important. Optimization of the latter (i.e., control hy gate A) may remove a connectivity matrix entry while the wire still remains. It can be seen that it is only necessary to be able to disable all leads that fit the definition of glohal feedbacks. This is a considerahly less stringent requirement than heing ahle to disahle *all* circuit-pack outputs.

The concept of logic disahling is a method of emulating the physical removal of circuit packs (i.e., leaving circuit-pack-output gates in the all 1's state). However, if the stuck-at-0 output of the gate is disahled, it presents a potential prohlem. This can be analyzed as follows. First, the stuck-at-0 output may feed to a higher-order node. If the ordering has been carefully observed, there must be an alternate method of controlling the lead. In this case the highest circuit pack fed hy the stuck-

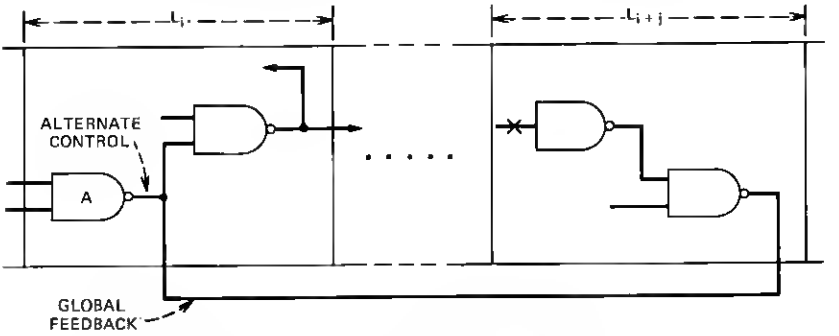


Fig. 8—Example of global feedback.

at-0 output is identified as bad. If ordering has not been carefully observed and no special control has been added, the result is unpredictable. If the stuck-at-0 output feeds only lower nodes and control or observation has not been added, the highest of the lower nodes are identified as bad. If observation has been added, the fault should be properly isolated. In the cases where the ordering has been observed, the pack identified as had is either the proper one or is fed by the stuck-at-0 output. A simple check is to remove the circuit pack identified by diagnosis. Next, all packs feeding the removed circuit pack are disabled. At this point, all connector pins should be logical 1's. A test connector can then he inserted into the vacant slot and examined. Assuming that the basic connectivity information is available on a per-pack basis, the actual faulty pack is now identified by correlating any grounded pins with a faulty pack.

The test procedure is highly automatic. Diagnosis proceeds to locate a suspect pack. The craftsman replaces this pack with the test connector. The machine disables the necessary packs, scans the connector, and locates any grounds. The location of the grounds can be combined with connectivity information stored on hulk storage to uniquely identify the bad circuit pack. This procedure is much less susceptible to manual errors than previous diagnostic techniques.

III. FEASIBILITY STUDY—APPLICATION OF COMET TO A SMALL PROCESSOR

To verify the feasibility of the COMET procedure, a small self-checking processor was selected for study. This choice was made he-cause a simulation model of the processor existed. This allows verification by simulating the ability of COMET to locate faults. In addition, the processor is complex enough to present a good sampling of "real-life" prohlems.

3.1 Brief description of processor

The processor is a stored program machine,[3] composed of approximately 4,400 logic gates. It features a microprogram control and a general-register structure. It is fully self-checking and does not rely on matching for fault detection. From a system view, the active and standby processors are linked to the outside world by the local maintenance center.[6] The LMC is responsible for the diagnosis of an off-line central control. In terms of COMET, control of the disahling and actual testing is exercised hy the LMC.

The interface hetween the processor and the LMC is detailed in Fig. 9. The major ports for controlling the processor are the input bus,

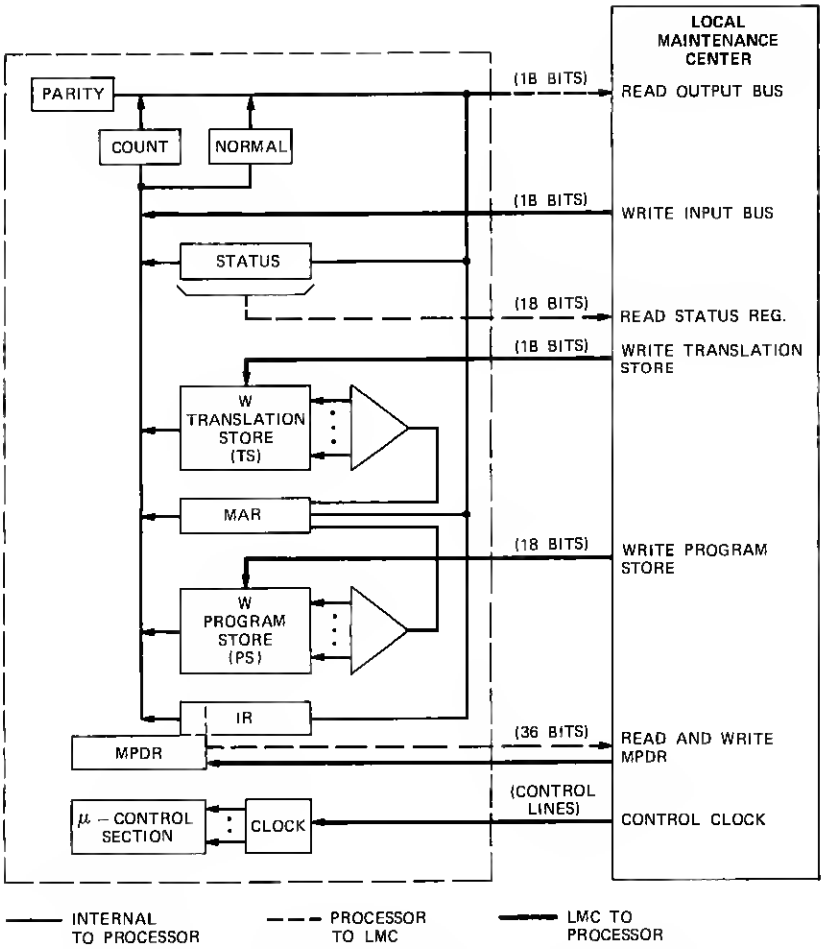


Fig. 9—Interface between processor and local-maintenance center.

the microprogram data register (MPDR), the clock, and tbe various stores. The ports prescribe external controllability and observability for tbe processor. Tbe internal structure of the processor is a register-bus structure witb 16 addressable registers.

The control of tbe machine centers around a microprogram unit. This unit is composed of a 36-bit data register and two major decoders, the "to" and "from" decoders. It is higbly self-checking as is tbe entire machine. Nearly any bardware error causes immediate notification of the LMC via tbe status register.[6]

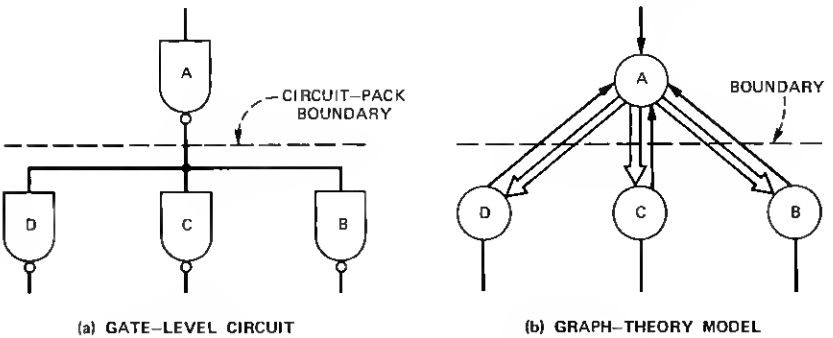
3.2 *Formation of the connectivity matrix*

Tbe first step in deriving tbe processor connectivity matrix is to define a reasonable set of functional nodes. In this case, the functional node definitions parallel the processor block diagram quite closely. There are a total of 41 functional nodes under consideration. These vary considerably in size. The LMC, for instance, is nearly as large as an entire processor wbile the decision logic is only a few gates.

With a preliminary set of functional nodes defined, tbe controllability and observability relations can be derived. These relations arc dcrived on a local (i.e., node-hy-node) hasis. In other words, the set of relationships for a single node can he written hy only considering its neighhoring nodes. In general, these relationships bear a closc association to the physical connections in a circuit. In fact, in the limit, every physical connection between two nodes could generate both a control and an observation edge.

Tbe mechanical construction of the connectivity matrix by only physical connection information will yield a sufficient set of conditions for leveling. This will be equivalent to doing a bit-by-bit OR of tbe control connectivity matrix witb its transpose to formulate tbe combined control and observation matrix. However, in practice, simplification can often be achieved without resorting to the seemingly hrute-force approach.

The method of simplification is based on two logic features not properly represented hy the graph-theory model. The first and most important feature is thc existence of fanout. A gate from a register may fan out to several points, as shown in Fig. 10a. The resultant directed graph for proper controllability and observability relations was derived strictly based on physical connection information



(a) GATE–LEVEL CIRCUIT (b) GRAPH–THEORY MODEL

Fig. 10—Fanout considerations in modeling.

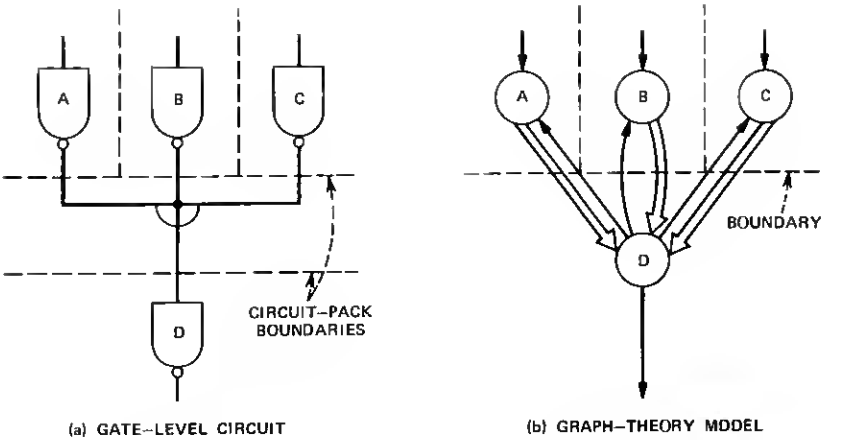


Fig. 11—Collector-tie considerations.

shown (see Fig. 10h). The interpretation is that all nodes B and C and D are required to observe node A. In fact, however, it is only necessary to have node B or C or D to observe node A. Rather than have entries in the observability matrix for all three, only one entry is needed. The determination of which entry to retain is usually not completely arbitrary. In general, it is desirable to retain only the highest-order node (B, C, or D) for observation.

The other logical feature that allows simplification is the collector tie (see Fig. 11a). This effect on the control-connectivity matrix is analogous to the effect of fanout on the observation-connectivity matrix. It is only necessary to be able to control one of the inputs to a collector tie to check its validity with respect to the gates feeding it (Fig. 11h).*

3.3 Ordering and partitioning of nodes

The 41 nodes may now be analyzed to arrive at a partial ordering. If they can be leveled at this point, an order is established. In general, this will not be the case. Analysis programs are then used to identify and locate controllability and observability MSCs. There are several ways to treat these problems. From a diagnostic point of view, the MSCs should be broken down into link-graph structures. This implies adding control or observation points and reanalyzing the graph. This

*The resolution of a ground on a collector-tied node is a well-known classical problem.

approach will yield a circuit that is diagnosable to one functional node (which may be one circuit pack or part of a circuit pack).

In practice, adding sufficient control or observation to break all MSCs may be expensive. In many cases the acceptance of reduced resolution is more attractive than the addition of much hardware. For example, it may not be economical to resolve a circuit pack output stuck-at-1 fault and an input-diode-open fault of the pack it drives, even if these two packs form a two-node MSC. Whenever possible one should always attempt to put the connected nodes on a single package to reduce (and eliminate) MSCs. This is equivalent to admitting that faults in the nodes are indistinguishable. However, this is of no importance if they are detectable and on a single package.

In attempting to partially order the processor nodes, several controllability and observability MSCs were discovered. The most obvious concern is the status register. The states of all check circuits in the machine are sampled and trapped in the status register. Any error indications cause an immediate maintenance interrupt of the processor. The LMC has the ability to directly read all bits of this register and to clear the register. However, it did not originally have controlled write access to the register. The situation that existed is shown in Fig. 12. This can be written as follows:

STATUS REGISTER → "TO" DECODER 1/N CHECK → "TO" DECODER ⇒ STATUS REGISTER

and indicates the presence of a loop. The flip-flops of the status register are not testable in a "start-small" diagnosis. The solution to this problem is rather simple. The LMC is given controlled write access to the status register, as shown in Fig. 12. This allows removal of tbe control link specifying:

"TO" DECODER ⇒ STATUS REGISTER

and replacing it with:

LMC ⇒ STATUS REGISTER.

In addition, there is a two-node loop between the MPDR and the microprogram store (MPS). This is solved by careful merging of the LMC access to the MPDR. The collector-tie access will restrict fault isolation to two packs (one in the LMC and one in the MPS) in the stuck-at-0 case but will allow resolution of the stuck-at-1 problems to one pack. The decision logic and MPDR are also connected by a loop. The decision logic controls bit 0 of the MPDR and is observed by the MPDR. The stuck-at-0 on the decision logic output would only

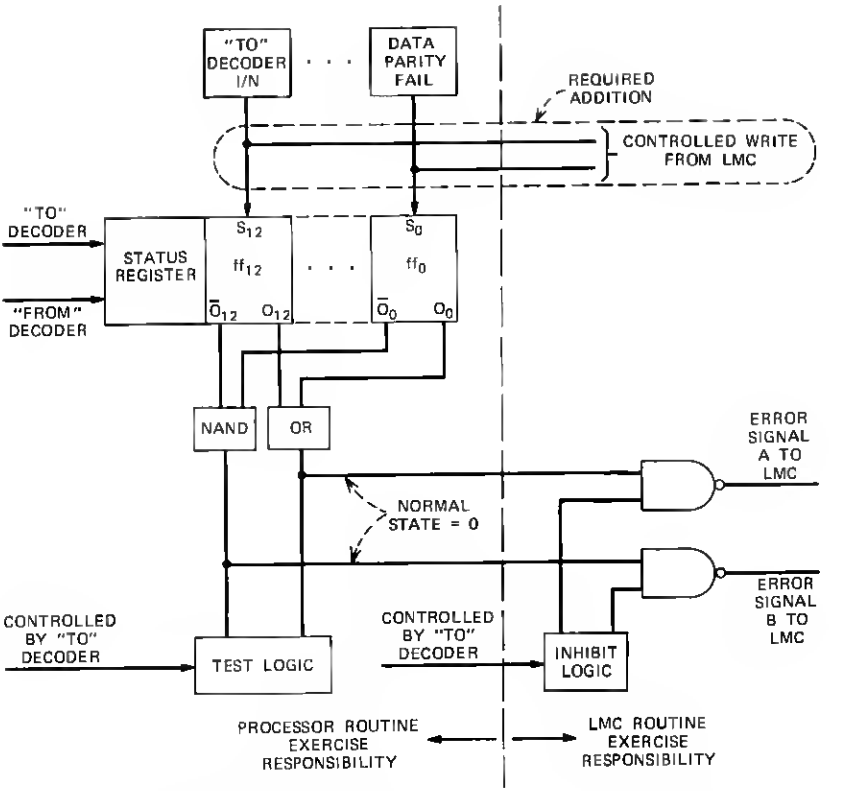


Fig. 12—Status register and its local-maintenance-center interface.

be resolvable to two packs. The stuck-at-1 output is not distinguishable from the set-bit-0 input open. The provision of a single control-write lead from the LMC eliminates the latter resolution problem.

Another loop exists between the decoders and the $1/N$ check circuits. Again, either two-pack resolution must be accepted or the two nodes must be packaged together if additional hardware is not provided. In this case, the decoder functions can be halved into an even parity and an odd parity and part of the check circuitry packaged with each half. The tradeoff again is one of engineering judgment and cost effectiveness.

The process of ordering the processor nodes can be completed by consciously resolving any conflict that appears. The term "consciously" is a key point. Assuming that the control and observation relations are complete, the analysis procedure will point out diagnostic

problems. To proceed to a fully ordered graph, these problems must he examined and treated. Thus, the existence of a fully ordered graph should insure consideration of diagnostic problems. A by-product of the ordering process will he the addition of maintenance hardware or a generated list of graph edges having some degree-of-resolution problems. The nature of this list of edges is such that it should give a reasonably good measure of the diagnostic resolution possible with the proposed design.

The final ordering of the functional nodes (Fig. 13) requires seven levels to encompass the 41 functional nodes. In this case there are fewer circuit packs than nodes.[3] This indicates that more than one node will he packaged on a circuit pack, as expected. The diagram gives some guidelines as to which nodes can conveniently be packaged together and which should not be packaged together.

From the ordered control/observation graph, a diagnosis strategy can be derived. Here a linear-partitioning process is used. In a linear-partitioning process, all circuit packs hut one of the highest order are first disabled. Tests are run on this circuit pack and, if a failure is detected, the fault is in the circuit pack. If the failure is not detected,

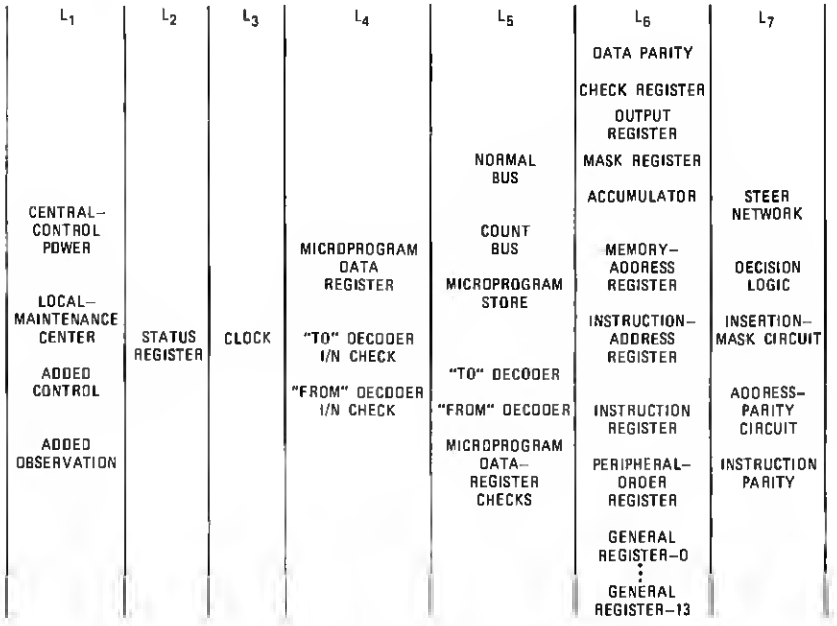


Fig. 13—Partial ordering of processor nodes.

further disabling is in order. In a second partition circuit, packs 1 and 2 (of order 1 and 2) are enabled, whereas the rest of tbe circuit packs are disabled. Tests are tben run on circuit packs 1 and 2. If they pass the tests, then the fault(s) would be in tbe rest of the disabled proportion; if they fail the tests, then the faulty circuit pack is in the portion of circuit under test. Since circuit pack 1 was verified to be good in the previous partition the faulty circuit would be circuit pack 2. This process is repeated for each partition in a linear fashion until the faulty pack is located.

In this example, the diagnostic strategy is to verify tbe first five levels in order. The power (L_1) is verified from the LMC. Next, the status register (L_2) is verified by controlled write and read access. Witb the status register now available as an observation port, the clock (L_3) can be verified. Next, controlled read and write access are used to verify the MPDR and the decoder check circuits (L_4). The MPDR and the status register provide sufficient control and observation to diagnose the MPDR check circuits (L_5). Cycling through the MPS and observing the results witb the MPDR will verify the microstore (L_5). The decoders (L_5) are now checked using tbe MPDR and clock as inputs and the status register as outputs. Note tbat tbe combination of tbe previously discussed packaging reasons could obviously modify L_5 and L_6. The buses (L_5) are verified by controlled read and write access. This leaves a skeleton processor capable of executing microinstructions. The nodes of L_6 and L_7 must be arbitrarily converted into a form adaptahle to linear enabling and diagnosis; an example is shown in Fig. 14. It was largely tbese functional nodes that were used for the simulation experiments.

3.4 Fault location using COMET

To verify the results tbat are expected from COMET, several simulation experiments were performed. A 2700-gate* simulation model of tbe processor was used. The model, which excluded the various stores, the peripheral communication and the LMC interfaces, provides a good vehicle for checking the COMET technique.

The procedure for checking the control section of the machine is quite straightforward with the aid of the LMC. Tbus, simulation was performed assuming that the microcontrol section had been tested and found good.

* The major difference in gate count between the simulation model (2700 gates) and the entire processor (4400 gates) is due to the inclusion of only two of the general-purpose registers in the simulation model.

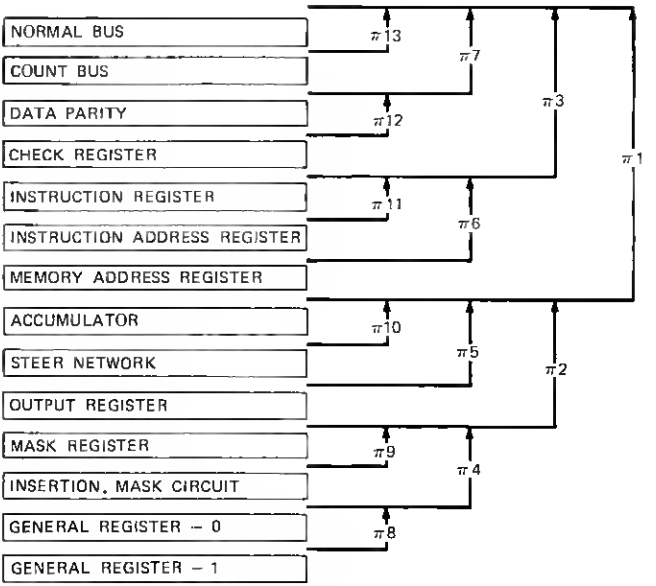


Fig. 14—Functional nodes included in simulation experiments.

Selected faults were inserted into the model and a simplified set of diagnostic tests were run. The functional node arrangement and the test (πi) procedure are shown in Fig. 14 in which a binary-partitioning technique is used. In actual applications, this would be replaced with a linear-partitioning arrangement.

The first experiment consisted of inserting a stuck-at-0 fault in bit 10 of the first level of the combinational data-rotation network.[3] The sequence of tests and their results, as derived by simulation, are shown in Table I. Tests $\pi 1$ and $\pi 10$ passed, whereas $\pi 2$ and $\pi 5$ failed. The pass/fail number of $\pi 1\ \pi 2\ \pi 5\ \pi 10 = (1001)$ uniquely points to a functional node which, in this case, is the rotation or "steer."

The second experiment was to insert a stuck-at-0 fault in the instruction address register (IAR). The results of running the test phases on this fault are shown in Table II. Again, a unique functional node is isolated by the pass/fail number $\pi 1\ \pi 3\ \pi 6\ \pi 11 = (0101)$.

For a third experiment, both the fault in the rotation logic and the fault in the IAR were inserted simultaneously. As expected, the IAR (i.e., the highest) fault was isolated independent of the other fault. Upon correction of the IAR fault, the steer fault would be isolated. If the replacement circuit pack that is to correct the IAR fault had been

Table I — Simulation results—steer network fault

Machine	Test (See Fig. 14)	Input Vector	Output Vector (Binary)
Good	$\pi 1$	# 7	0000 0000 0000 0000 00
Faulty	$\pi 1$	# 7	0000 0000 0000 0000 00
Good	$\pi 1$	#12	1111 1111 1111 1111 11
Faulty	$\pi 1$	#12	1111 1111 1111 1111 11
Good	$\pi 1$	#14	0000 0000 0001 0000 11
Faulty	$\pi 1$	#14	0000 0000 0001 0000 11
Good	$\pi 1$	#16	1000 0000 0001 0000 10
Faulty	$\pi 1$	#16	1000 0000 0001 0000 10
Good	$\pi 2$	# 8	0000 0000 0000 0000 00
Faulty	$\pi 2$	# 8	0000 0000 0010 0000 00
Good	$\pi 2$	#13	1111 1111 1111 1111 11
Faulty	$\pi 2$	#13	1111 1111 1111 1111 11
Good	$\pi 5$	# 6	0000 0000 0000 0000 00
Faulty	$\pi 5$	# 6	0000 0000 0010 0000 00
Good	$\pi 5$	# 9	1111 1111 1111 1111 11
Faulty	$\pi 5$	# 9	1111 1111 1111 1111 11
Good	$\pi 10$	# 6	0000 0000 0000 0000 00
Faulty	$\pi 10$	# 6	0000 0000 0000 0000 00
Good	$\pi 10$	# 9	1111 1111 1111 1111 11
Faulty	$\pi 10$	# 9	1111 1111 1111 1111 11

faulty itself, it would also have been isolated. This demonstrates the capability of isolating multiple faults.

The final simulation experiment involved altering the basic IAR circuit to reflect a signal short between bits 4 and 14 of the register. The test set used has sufficient fault-detection capability to recognize this nonclassical fault. The COMET approach makes it possible to isolate the fault, as indicated by the simulation results shown in Table III.

These experiments bear out the results expected for isolating single, multiple, and/or nonclassical faults. They also point out the dependence on a good set of tests. The signal short is an example of this dependence. Normally, a set of tests capable of detecting all stuck-at-1 or stuck-at-0 faults might not detect the short. However, if the fault is detectable, it is isolated by using COMET. Considerations such as these will obviously have some impact on the diagnostic program design.

IV. TRADE-OFFS

Most of the discussion thus far has considered diagnostic resolution to a single circuit pack. If COMET dictated an all-or-nothing approach to the problem, its usefulness would be severely affected. The cost of

Table II — Simulation results—instruction address register classical fault

Machine	Test (See Fig. 14)	Input Vector	Output Vector (Binary)				
Good	$\pi 1$	# 7	0000	0000	0000	0000	00
Faulty	$\pi 1$	# 7	0000	1000	0000	0000	00
Good	$\pi 1$	#12	1111	1111	1111	1111	11
Faulty	$\pi 1$	#12	1111	1111	1111	1111	11
Good	$\pi 1$	#14	0000	0000	0001	0000	11
Faulty	$\pi 1$	#14	0000	1000	0001	0000	11
Good	$\pi 1$	#16	1000	0000	0001	0000	10
Faulty	$\pi 1$	#16	1000	1000	0001	0000	10
Good	$\pi 3$	# 4	0000	0000	0000	0000	00
Faulty	$\pi 3$	# 4	0000	0000	0000	0000	00
Good	$\pi 3$	# 5	1111	1111	1111	1111	11
Faulty	$\pi 3$	# 5	1111	1111	1111	1111	11
Good	$\pi 3$	# 7	0000	0000	0001	0000	11
Faulty	$\pi 3$	# 7	0000	0000	0001	0000	11
Good	$\pi 3$	# 8	1000	0000	0001	0000	10
Faulty	$\pi 3$	# 8	1000	0000	0001	0000	10
Good	$\pi 6$	# 6	0000	0000	0000	0000	00
Faulty	$\pi 6$	# 6	0000	1000	0000	0000	00
Good	$\pi 6$	#10	1111	1111	1111	1111	11
Faulty	$\pi 6$	#10	1111	1111	1111	1111	11
Good	$\pi 11$	# 5	0000	0000	0000	0000	00
Faulty	$\pi 11$	# 5	0000	0000	0000	0000	00
Good	$\pi 11$	# 8	1111	1111	1111	1111	11
Faulty	$\pi 11$	# 8	1111	1111	1111	1111	11

removing *all* control/observation MSCs is in general quite high. There are a number of alternatives that can and should be evaluated.

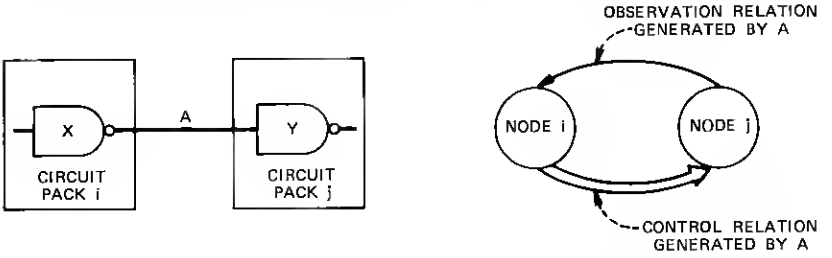
First, packaging must be considered as a trade-off parameter. It is possible that toleration of a slight reduction in packaging density could result in a significant reduction in the number of MSCs left to be broken. Procedures could be derived to evaluate the possible packaging trade-offs, but for now it appears to be a matter of engineering judgment.

Second, the possibility of accepting reduced resolution must also be considered. A vast majority of the loops in the controllability and observability connectivity matrix contain leads going from one pack to another. An example of those leads is shown in Fig. 15a. The *wire* marked A will generate a control relation and an observation relation, as shown in Fig. 15b. COMET analysis of this situation will reveal that to distinguish between an output stuck-at-1 fault (on gate X) and an input-diode-open fault (on gate Y), one must add control or

Table III — Simulation results—instruction address register nonclassical fault

Machine	Test (See Fig. 14)	Input Vector	Output Vector (Binary)				
Good	$\pi 1$	# 7	0000	0000	1111	1111	11
Faulty	$\pi 1$	# 7	0000	1000	1111	1111	11
Good	$\pi 1$	#12	1111	1111	0000	0000	11
Faulty	$\pi 1$	#12	1111	1111	0000	0010	11
Good	$\pi 1$	#15	0000	0000	0001	0000	11
Faulty	$\pi 1$	#15	0000	0000	0001	0000	11
Good	$\pi 1$	#19	1000	0000	0001	0000	10
Faulty	$\pi 1$	#19	1000	0000	0001	0000	10
Good	$\pi 3$	# 4	0000	0000	1111	1111	11
Faulty	$\pi 3$	# 4	0000	0000	1111	1111	11
Good	$\pi 3$	# 5	1111	1111	0000	0000	11
Faulty	$\pi 3$	# 5	1111	1111	0000	0000	11
Good	$\pi 3$	# 8	0000	0000	0001	0000	11
Faulty	$\pi 3$	# 8	0000	0000	0001	0000	11
Good	$\pi 3$	#10	1000	0000	0001	0000	10
Faulty	$\pi 3$	#10	1000	0000	0001	0000	10
Good	$\pi 6$	# 6	0000	0000	1111	1111	11
Faulty	$\pi 6$	# 6	0000	0000	1111	1111	11
Good	$\pi 6$	#10	1111	1111	0000	0000	11
Faulty	$\pi 6$	#10	1111	1111	0000	0000	11
Good	$\pi 11$	# 5	0000	0000	1111	1111	11
Faulty	$\pi 11$	# 5	0000	1000	1111	1111	11
Good	$\pi 11$	# 8	1111	1111	0000	0000	11
Faulty	$\pi 11$	# 8	1111	1111	0000	0010	11

observation to line A. However, if two-pack resolution is tolerable, one can ignore this two-node loop generated by wire A. Experience has shown that a very large number of the problems that graph-theoretic analysis points out are of this type. It is reasonably simple to handle these problems if reduced resolution is tolerable. When



(a) CIRCUIT REPRESENTATION

(b) CONTROL/OBSERVATION REPRESENTATION

Fig. 15—Resolution vs cost trade-off.

COMET analysis points out one of these loops, the engineer can remove either the control or observation entry in the connectivity matrix and place it in a reduced resolution list. Later, after leveling the grapb, he will go back and determine a set of suspect packs to be identified when a pack is found to be bad. Tbus, the most probably faulty pack is named along witb packs that could have stuck-at-1 outputs. Tbe information to do this is available in the previously created reduced-resolution list.

It is likely that "hybrid" techniques are also possible. A small number of faults that would be costly to handle by COMET could be fault simulated. The simulation of this small set of faults is much less expensive than the totally fault simulated case. This approach, bowever, is open to tbe same criticism as the traditional exact-matcb TLM. Consistent results may be hard to come by and one also must update the fault simulation as any changes are made.

COMET is obviously of most value wben applied completely. Engineering considerations may point to this being undesirable. In this case, a number of trade-offs and less costly variations on COMET can be used.

V. IMPACT OF COMET

5.1 Design for maintainability

The major impact of COMET should be a partial redefinition of design philosophies for digital systems. COMET is, after all, an *engineering technique.* It is capable of identifying diagnostic resolution problems before the design has been frozen and allows the design engineer to consciously evaluate the possible problems. To proceed in the analysis, the engineer must determine what is to be done about the problem. His decision may lead to the compilation of a table-of-resolution problems. This list can give an idea of the overall fault resolution possible for a design. Iteration of the process will allow full evaluation of the cost/resolution trade-off before tbe design is committed to hardware. With this tool, diagnosability takes its place as a primary design parameter with a reasonably well-defined set of trade-offs for evaluation.

Efficient application of COMET depends in part on the orderly application of the tecbnique. Analysis should first consider ratber global functional blocks. The information gained at this stage of analysis provides the basis for optimization of the control and observation relations. The global analysis will usually determine which of the observation edges to retain. It can also help prescribe necessary

external control and observation of the system. In many cases, the ability to provide only necessary external access can lead to minimization of the interaction of systems and associated sanity-preservation problems.

Once the global analysis phase has been completed, the expansion of the global functional nodes on a local (node-by-node) basis can take place. The task is to expand the global functional node into identifiable entities. COMET is then used within the global node to specify an organization that is diagnosable. The problem has heen converted to one of modularization. By carefully considering the amount of connectivity of nodes, packaging problems can also be anticipated.

One outgrowth of the proposed design approach should he an increased awareness of the functional aspects of a system. This, in turn, should lead to more attempts at functional testing and an increased ability to do localized test design. COMET requires conscious consideration of test design and conscious consideration of the controllahility and observahility of an entity. It further requires that diagnosis he ordered such that these input/output ports are verified prior to testing the entity in question. This information may be sufficient to allow test design and verification on a local hasis. This would lend itself to algorithmic test generation and inexpensive verification. The ordering process, if adhered to, could also substantially reduce the problem of having to simulate faults in large blocks of circuitry for test-design purposes.

Probably a more important aspect of the functional orientation of diagnosis is the emphasis on detection. Traditional diagnostic design relies on detection of all single stuck-at types of faults. In integrated-circuit technology, this may be a somewhat restricted subset of all possible failures. The nonclassical failures may cause trouble in traditional diagnosis, even though they are nearly always detectable. With COMET, detection is the only thing that is important.

5.2 Other applications of COMET

COMET has a number of advantages over normal diagnostic procedures. In the simulation experiments, the ahility to diagnose multiple independent faults was touched upon hriefly. This ability means that a machine designed using COMET can be tested upon installation with the regular diagnostic program. The procedure would be to insert a selected number of the highest-order nodes and run the diagnostic. The missing packs appear to be disabled. The initial packs can be diagnosed and, if no faults are found, the process can be con-

tinued. Otherwise, the faulty pack or packs are replaced and the process repeated. The ahility to isolate nonclassical faults was also touched upon briefly. The impact of this on the diagnostic program must he evaluated.

It should be noted that COMET can he used at various levels of design and analysis. For instance, the first analysis may assume functional nodes to he of the suhsystem size. At this level of detail, the overall system diagnostic philosophy may he evaluated and improved. Next, each functional node of suhsystem size may he suhdivided into functional nodes of circuit-pack size. Assuming that the circuit packs may eventually be composed of LSI chips on a ceramic, it may be feasihle to consider subdividing the circuit-pack-sized functional nodes into LSI-chip-sized functional nodes. The potential henefit here is in the area of factory repair. If COMET is applied at the chip level it becomes possihle to isolate the faulty LSI chip using factory test apparatus. Faulty chip location is done hy automatically disabling chip outputs and testing the remaining chips. Conceptually, this could simplify the problems of repair of LSI packages.

5.3 Further work

There are several aspects of COMET that will require further work. Most apparent is work intended to automate a large part of the analysis and synthesis phases. This work appears to be of an open-ended type. A system that iteratively seeks a near optimum solution while considering physical design and other constraints would need to he a sophisticated system. The system of program aids that has heen implemented as part of LAMP represents a start in this direction.

In addition, some thought must also be directed to obtaining the functional node definitions. These do not of necessity have to parallel the function performed, hut can reflect such techniques as hit slicing and other packaging considerations.

The idea of relying on routine exercise procedures to check failures in logic disahling is not very appealing. It is possihle that improvements can he made in this area.

VI. SUMMARY

The concept of controllability and ohservahility for organized system design to enhance diagnosahility has heen introduced. It makes use of logic disahling of circuit modules (or packs) and the proper ordering of functional nodes of a system. Both hinary and linear ordering can be used. Other arrangements are possihle hut have not

been investigated. Graph-theoretic algorithms for determining the optimum placement of control and access and monitor points for diagnostic testing were presented.

A feasibility study was performed by applying COMET to an existing processor. The capability of locating single, multiple, and non-classical faults was demonstrated, based solely on the pass/fail indication of test partitions. Additional control and monitor circuitry was added to satisfy the requirements of COMET. The added hardware, which included both the logic for disabling and for modifications is modest (e.g., less than 10 percent).

For a given set of tests, significant savings in simulation time to generate TLM data are achievable for a design using COMET. Furthermore, if a laboratory or a field model of the system is available, the TLM data generation effort will be almost totally eliminated. Data can be generated by actually running the diagnostic program on the machine for which a TLM is to be produced. This approach also makes TLM update easy and therefore drastically reduces many problems caused by machine differences and circuit changes. Finally, it offers the possibility to tailor-make TLMs that can be stored on-line.

Application of COMET is best suited for new designs where the trade-offs among circuit design, logic packaging, and diagnosability can be jointly considered early in the design stage. The use of machine aid tools would make this feasible and could greatly facilitate the design process. Retrofiting COMET to existing systems may be feasible on many designs, but may not be economical on others. Engineering judgment must be exercised to study the impact.

REFERENCES

1. H. Y. Chang, E. G. Manning, and G. Metze, *Fault Diagnosis of Digital Systems*, New York: John Wiley and Sons, 1970.
2. IEEE Trans. on Computers, Special Issue on Fault-Tolerant Computing, *C-22*, No. 11 (November 1971), pp. 1241–1435.
3. H. Y. Chang, R. C. Dorr, and D. J. Senese, "The Design of a Microprogrammed Self-Checking Processor of an Electronic Switching System," IEEE Trans. on Computers, *C-22*, No. 5 (May 1973), pp. 489–500.
4. C. V. Ramamoorthy, "A Structural Theory of Machine Diagnosis," AFIPS Conference Proceedings, 1967 Spring Joint Computer Conference, *30*, 1967, pp. 743–756.
5. C. V. Ramamoorthy and L. C. Chang, "System Segmentation for the Parallel Diagnosis of Computers," IEEE Trans. on Computers, *C-20*, No. 3 (March 1971), pp. 261–270.
6. H. Y. Chang, G. W. Heimbigner, D. J. Senese, T. L. Smith, "Maintenance Techniques of a Microprogrammed Self-Checking Control Complex of an Electronic Switching System," IEEE Trans. on Computers, *C-22*, No. 5 (May 1973), pp. 501–512.